Title: Wrong Turns on a Roundabout Roadway

Brief Overview:

In England, there is an infamous double roundabout roadway (see Figure 1). On this roadway we define a wrong turn to be any turn that results in a longer trip than necessary, and we assume that every time a wrong turn is possible, the probability of making a wrong turn is 1/2. We are interested in the average number of wrong turns made during one trip through the maps which will consist of one, two, three, four, and five consecutive roundabouts (see Figure 2 for the three roundabout picture).

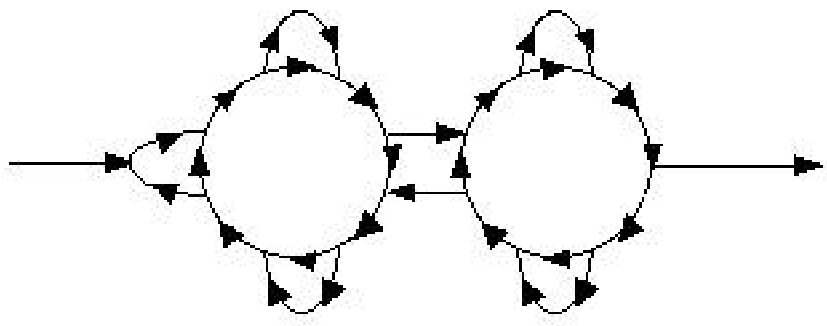


Figure 1: Double roundabout roadway

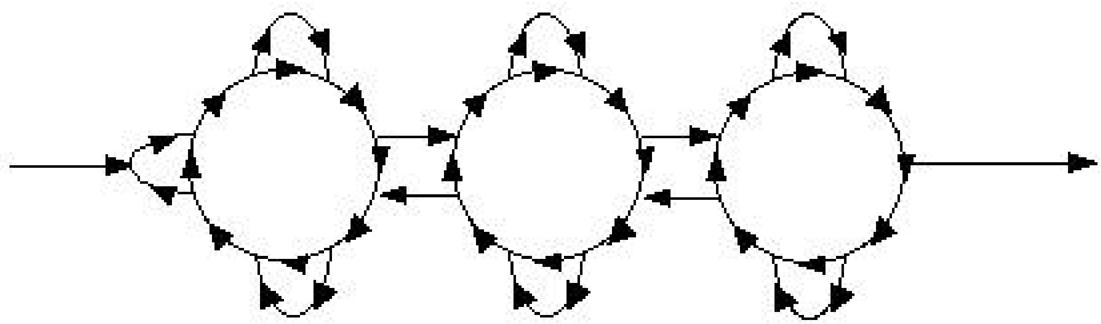


Figure 2: Triple roundabout roadway

Students are expected to have a solid background in both algebra and computer programming (e.g., we used the the C programming language, but any other language is fine). They will apply C programming skills to writing a computer program that calculates the average number of wrong turns made given the number of roundabouts in a row. They will calculate this number by simulating 1000 trips through the map. Students will then attempt to generalize their solutions by deriving a formula that expresses their findings in algebraic terms. By doing this, they will have generalized the solution so that the average number of wrong turns for any number of consecutive roundabouts can be calculated without running the simulation.

This problem involves mathematical modeling which is used to accomplish many varied tasks in the real-world. Examples of real-world situations that require mathematical modeling include:

- scheduling of class times and locations at a particular school
- timing of traffic lights in a city

- analysis of financial markets
- simulations used in chemistry and physics
- the scheduling of planes coming into and leaving a particular airport

In addition to this, the idea of simulating an event thousands of times in order to be able to make predictions regarding a particular situation is a technique that is often used when a mathematical formula is not known, yet the results of such a formula would be useful.

Finally, the students will gain necessary experience in writing about a technical topic and presenting a technical briefing. Both of these skills are called upon in the real-world.

Links to NCTM 2000 Standards:

- **Mathematics as Problem Solving**
  Students will be presented with a research problem for which the answer is not immediately obvious. They will write a C computer program to simulate travel on a roundabout roadway in order to find the average number of wrong turns given the number of roundabouts.

- **Mathematics as Reasoning and Proof**
  Students will brainstorm about possible ways a computer program can be used to solve the given problem. They will then organize their ideas in order to convert their best ideas into an algorithm that can be implemented in a computer program. Finally, in writing their computer programs, students will use reasoning skills to define exactly what the computer must do in order to simulate travel on the roadway. Reasoning skills will come into play again when students both look for errors in their programs and then determine how to fix these errors.

- **Mathematics as Communication**
  Students will document their work in a technical paper. This paper will contain a brief but clear description of the original problem, a summary of how they attempted to solve the problem, and their final results and conclusions. They also will present their findings to their peers in an oral presentation that will summarize their technical paper.

- **Mathematics as Connections**
  Students will apply a variety of topics in mathematics along with computer skills in order to complete this project. These topics include algebra, elementary statistics and probability, mathematical modeling, and logic. They also will gain experience in technical writing, as well as giving a technical presentation, both of which are essential in the real-world.

- **Patterns, Functions, and Algebra**
  Students will search for patterns among expected numbers of wrong turns. Based on the patterns they find, they will formulate an algebraic equation which will summarize their findings.

- **Data Analysis, Statistics, and Probability**
  Students will use the elementary statistical notion of averages in order to calculate the average number of wrong turns. Elementary probability will also be used in determining how often and when a wrong turn is made.

  Once the students have generated an average for each map, the single, double, triple, and quadruple roundabouts, they will analyze the relationships between the averages in order to find a formula that can be used to calculate the average number of wrong turns for any number of consecutive roundabouts without running the simulation.

Links to Maryland High School Mathematics Core Learning Goals:

Functions and Algebra

- **1.1.1**
  Students will search for patterns among average numbers of wrong turns. Based on the patterns they find, they will formulate an algebraic equation which will summarize their findings.

Geometry, Measurement, and Reasoning

- **1.1.1**
  Students will brainstorm about possible ways a computer program can be used to solve the given problem. They will then organize their ideas in order to convert their best ideas into an algorithm which will be implemented in a computer program. Finally, in writing their computer programs, students will use reasoning skills to define exactly what the computer must do in order to simulate travel on the roadway. Reasoning skills will come into play again when students both look for errors in their programs and then determine how to fix these errors.

Data Analysis and Probability

- **1.1.1**
  Students will use the elementary statistics notion of averages in order to calculate the average number of wrong turns. Elementary probability will also be used in determining how often and when a wrong turn is made.

  Once the students have generated an average for each map, the double, triple, and quadruple roundabouts, they will analyze the relationships between the averages in order to find a formula that can be used to calculate the average number of wrong turns for any number of consecutive roundabouts without running the simulation.

Grade/Level:

Grades 9-12

Duration/Length:

2 weeks of 90-minute classes (with both in-class and outside-of-class work)

Prerequisite Knowledge:

Students should have working knowledge of the following skills:

- Algebra
- Computer programming in C

Student Outcomes:

Students will:

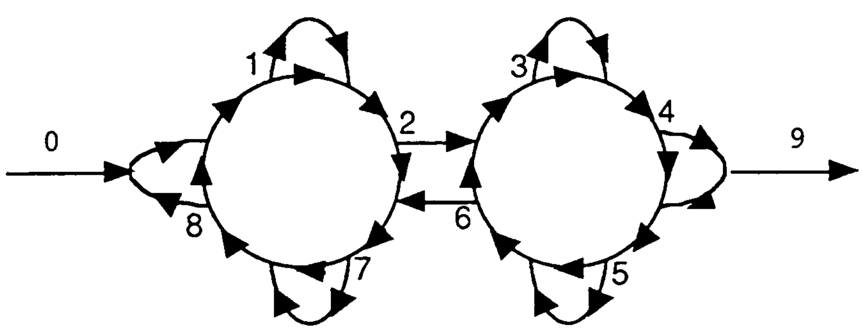
- write computer programs in C to simulate the problem.
- test their programs using a known solution.
- run their programs using various maps.
- look for patterns in the solutions produced by their programs.
- express patterns found using mathematical equations.
- document their findings in a technical paper.
- present their findings to a group of their peers in a clear and concise manner.

Materials/Resources/Printed Materials:

- Computer
- C compiler
- Word processor
- Paper
- Pencil

Development/Procedures:

Working in pairs, the students will write computer programs to approximate the average number of wrong turns made in various configurations of roundabouts. They will experiment using roundabouts consisting of 2 circles, 3 circles and 4 circles. In each case they will run at least 1000 simulations and compute the average number of wrong turns. These results will give accurate approximations to the true averages. The students will then attempt to find a formula for the average number of wrong turns on a roundabout with n circles. The correct formula is $n(n+2)$. As part of their research, the students should be strongly encouraged to pursue at least one of the problems in the Extension/Follow Up section. The students will then document their work in a brief paper. Finally, each team of students will present their work to their peers who have not done the same problem.



The diagram above is a roundabout that consists of 2 circles. The arrow at the left is the starting point and the arrow at the right is the ending point. Each of the numbered locations on the roundabout is a node. At each node, the traveler has a choice of directions. The traveler wants to go through the roundabout making as few wrong turns as possible. A wrong turn is a turn that makes a trip through the roundabout longer. On the diagram above, the nodes are numbered 0-9. For our purposes, we also will include the starting point as well as the ending point in our set of nodes. We will name the starting point node 0 and the ending point node 9.

We are now ready to begin. We want to simulate movement through the double-roundabout. One way to do this is to use two 2-dimensional matrices, a movement matrix and an error matrix. The movement matrix will indicate the number of possible ways to go from one node to the next node. For example, in the diagram, there are 2 ways to go from node 1 to node 2 but there is only one way to go from node 2 to node 3. All of the possible ways to go from one node to another can be summarized in a movement matrix.

In this matrix, we will use the array element movement[i][j] to store the number of possible ways to get from node i to node j. So far, we have movement[1][2]=2 and movement[2][3]=1. Continuing in this way, ask the students to construct the rest of the movement matrix.

Next, we need to construct an error matrix. Recall that there were 2 ways to move from node 1 to node 2. Taking one of those ways increases the length of the entire trip so we will consider this path to be wrong. So, from node 1 to node 2, there is one possible wrong turn. In our matrix, we will say error[1][2]=1. When moving from node 2 to node 3, however, there is only one possible way to move between the nodes so it must be correct. In our matrix, we say error[2][3]=0. Continuing in this way, error[*i*][*j*] contains the number of possible wrong turns from node *i* to node *j*. Ask the students to construct the rest of the error matrix.

Here are the completely filled in matrices:

Movement matrix

	0	1	2	3	4	5	6	7	8	9
0‖	0	1	0	0	0	0	0	0	0	0
1‖	0	0	2	0	0	0	0	0	0	0
2‖	0	0	0	1	0	0	0	1	0	0
3‖	0	0	0	0	2	0	0	0	0	0
4‖	0	0	0	0	0	1	0	0	0	1
5‖	0	0	0	0	0	0	2	0	0	0
6‖	0	0	0	1	0	0	0	1	0	0
7‖	0	0	0	0	0	0	0	0	2	0
8‖	0	2	0	0	0	0	0	0	0	0

Error matrix

	0	1	2	3	4	5	6	7	8	9
0‖	0	0	0	0	0	0	0	0	0	0
1‖	0	0	1	0	0	0	0	0	0	0
2‖	0	0	0	0	0	0	0	1	0	0
3‖	0	0	0	0	1	0	0	0	0	0
4‖	0	0	0	0	0	1	0	0	0	0
5‖	0	0	0	0	0	0	1	0	0	0
6‖	0	0	0	0	0	0	0	1	0	0
7‖	0	0	0	0	0	0	0	0	1	0
8‖	0	1	0	0	0	0	0	0	0	0

If we wanted to use either matrix, node *i* is the *i*th row and node *j* is the *j*th column. In order to verify that the students understand how to use the matrices, ask them to find movement[3][4] and error[3][4]. The answers should be movement[3][4]=2 and error[3][4]=1.

Once the students are able to generate these two matrices, the next step is to use them to simulate movement. To do this, they will start at node 0. Looking at the movement matrix, the only node they can go to is node 1 (movement[0][1]=1 and the rest of row 0 contains only 0s).

Next look at the error matrix to see how many wrong ways there are to get from node 0 to node 1. Error[0][1]=0 so that means there are no wrong ways to get from node 0 to node 1. So the students can move directly to node 1.

Since we now want to move to node 2, we again use the movement matrix. We can see that there are two ways to get from node 1 to node 2. Using the error matrix, we can see that there is one inefficient way to get from node 1 to node 2. Since both paths between the nodes are equally likely, we will have to randomly select a path. A fair way to do this is to flip a coin. If we get a “head”, we take one path and if we get a “tail” we take the other path. Either way, we end up at node 2.

Encourage students to continue to move through the entire roundabout until they get to the end, node 9. At each step along the way, they should keep track of how often they took a wrong turn. When they get to the end, they should write down the total number of wrong turns they took.

Once students have done this exercise by hand, suggest to them that now they can do it 999 more times. When the groaning stops, suggest that an alternate solution would be to write a computer program to simulate what they just did 999 times.

The only difference between manually simulating a trip through the roundabout and automating a simulation is the choice of which path to take when there are two possibilities. When writing the program, students should use a random number generator that selects a number between 0 and 1. Once such a number has been selected, the program can take one route if the number is less than 1/2 or the other route if the number is greater than or equal to 1/2.

The goal of writing the program is to simulate 1000 trips through the roundabout. Each time, students should keep track of the number of wrong turns takes. When 1000 trips have been made through the roundabout, the program should calculate the average number of wrong turns for a trip. To do this, they must sum up all of the wrong turns and then divide the sum by 1000. This will provide the average number of wrong turns for the given roundabout.

After the students have successfully programmed this problem, change the roundabout from 2 circles to 3 circles. This change is very easy to make and only involves changing the movement and error matrices. Remember that when you assign another roundabout, you will need to label the nodes. The names of the nodes don't matter - they just need to be named. Once you have another diagram ready, ask students to produce the average number of wrong turns for this new roundabout.

When they finish this, students should produce the average number of wrong turns for 4 circles.

They will now have the average for 2, 3, and 4 consecutive circles. The next step is to see if there is a pattern in the averages. If the students do not find a pattern, encourage them to program 5, 6, and 7 circles and then look for a pattern. The more averages the students have, the easier it will be to find the pattern.

Once students find the pattern, ask them to write a formula to express the pattern. The correct formula is:

$$n*(n+2)$$

where n is the number of circles in the picture.

If students have trouble writing the computer program, they can use the attached program as a guide for the 2-circle case. Also, in order to debug their programs, the students will need to know whether their results are reasonable. A good way to accomplish this is to give them the answer to the 2-circle case. Since the general formula is $n*(n+2)$, for n = 2 the answer is 8.

After completing the program and finding the formula, students must document their findings in a technical paper. The paper should include a description of the problem, a description of the computer program, the results of the program and the formula found. In addition to this, students should present their papers to other students who have not done the same problem.

Assessment:

Students will get three scores: one for the program, one for the paper, and one for the presentation. The scores for all three will be added together to produce a final score for the entire project:

Rubric for scoring the program:

12 points	If the program produces the correct answers and is well documented
8 points	If the program either does not produce correct answers or it is not well-documented
4 points	If the program neither produces correct answers nor is it well-documented
0 points	If no program was written

Rubric for scoring the paper:

The presentation should include each of the following:

Problem description
Program description
Results of program
Formula for the expected number of wrong turns given the number of consecutive roundabouts

Each section of the four sections of the presentation should be assigned a score ranging from 0-3 based on whether the section is present and how clearly and correctly it is documented in the paper (the scoring rubric is indicated below). Once each section receives a score, the scores in all sections are added to produce the final score for the paper.

3 points	If the section is present, clear and correct
2 points	If the section is present in the paper, and either not clear or not correct
1 points	If the section is present in the paper, and it is both not clear and not correct
0 points	If the section is not present in the paper

Rubric for scoring the presentation:

The presentation should be scored using the same criteria as the paper (see the instructions for scoring the paper above).

Once the program, paper, and presentation are all assigned scores, add up the three scores and this will produce a total score ranging from 0 - 36. The following list indicates what letter grade gets assigned to each numerical score:

A	32 - 36
B	28 - 31
C	25 - 27
D	21 - 24
F	0 - 23

Extension/Follow Up:

Students with the mathematical background required to sum geometric series could attempt to derive the formula $n*(n+2)$ for the expected number of wrong turns on a roundabout roadway having n circles. To accomplish this, they would first want to derive the result for a single circle, then 2 circles and so on. An induction proof can then be used to establish the formula $n*(n+2)$. Another extension would be to change the probability of a wrong turn to p, instead of 1/2, and repeat the original computer experiments. It would also be interesting to change the number of loops attached to each circle. This would change the number of places where a wrong turn can occur. The students could also create roundabouts consisting of different configurations of circles. For example, the circles could be laid out in an n by m grid instead of a straight line. There is an endless variety of such roundabout configurations that could be tested. Changing the interconnections between the circles is another possible modification. An extension that could be applied to the original configuration---or any of the previous extensions---is to allow at most one wrong turn onto each route. In this case, the answer for a roundabout consisting of 2 circles is 23/4.

Authors:

Didon Pachner
National Security Agency
Ft. Meade, Maryland

Brian Pilz
National Security Agency
Ft. Meade, Maryland

Mark Stamp
National Security Agency
Ft. Meade, Maryland

TEACHER NOTES

Below is a sample C program which simulates travel through a double roundabout roadway and calculates the expected number of wrong turns given that the probability of making a wrong turn at each node is 1/2. Students do not need to reproduce this program identically and other algorithms to do the job exist. This is just one possible way to do the calculation.

```
//
//  program simulates stepping through a 'roundabout'.
//  each roundabout consists of several circles which may
//  be departed from with a certain probability.  The simulation
//  will progress through the puzzle until escaping from the other
//  side, counting the number of wrong turns made.
//

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#define FINAL 9            // final position – changes with matrices
#define MAX_STEP 100 // maximum number of steps through graph to use
#define NOT_USED –1   // initialization value for columns in read
#define TRIALS 1000   // number of trials to use for simulation

main()
{
   int movement[FINAL+1][FINAL+1], // gives next states from current
      error[FINAL+1][FINAL+1];          // tells erroneous state
   int i,j,n,total,step,wrong_turn,current_node,non_zero,
      column1,column2,wrong,next_node,coin_flip;
   float average;

   // zero matrices //
   for ( i=0; i<=FINAL; i++)
   for ( j=0; j<=FINAL; j++){
      movement[i][j]=0;
      error[i][j]=0;
   }

   clrscr();

   // assign matrices //
   movement[0][1]=1;
   movement[1][2]=2;
   movement[2][3]=1;
   movement[2][7]=1;
   movement[3][4]=2;
   movement[4][5]=1;
   movement[4][9]=1;
   movement[5][6]=2;
   movement[6][3]=1;
   movement[6][7]=1;
   movement[7][8]=2;
   movement[8][1]=2;
```

```
error[1][2]=1;
error[2][7]=1;
error[3][4]=1;
error[4][5]=1;
error[5][6]=1;
error[6][7]=1;
error[7][8]=1;
error[8][1]=1;

// outer loop – number of trials //
total=0;
for( n=0; n<TRIALS; n++){

step = 0;
wrong_turn=0;
current_node=0;

// inner loop – step through graph //
while( current_node!=FINAL && step<MAX_STEP ){

non_zero=0;

// read first matrix information //
column1 = NOT_USED;
column2 = NOT_USED;
for( i=0; i<=FINAL; I++){
if(movement[current_node][i]!=0){
non_zero++;
if(column1==–1) column1 = I; else column2=I;
}}

// read second matrix information //
wrong = NOT_USED;
for( i=0; i<=FINAL; I++){
if(error[current_node][i]!=0) wrong = I;
}

// interpret information read //
if( non_zero==1 && movement[current_node][column1]==1 ){
next_node = column1;
} else {
if( column2==NOT_USED ){
next_node = column1;
coin_flip = rand() % 2;
if( coin_flip==1 ) wrong_turn++;
} else {
coin_flip = rand() % 2;
if( coin_flip==1 ){
wrong_turn++;
next_node = wrong;
} else {
if( column1==wrong ) next_node = column2;
else next_node = column1;
}}}
```

```
// update from result //
current_node = next_node;
step++;
}

total += wrong_turn;
}

// tell final result //
average = total / (float) TRIALS;
printf("total=%d average=%f\n",total,average);
return 0;
}
```